A Deep Learning Surrogate Framework for High-Dimensional Regression Problems in Mechanical Engineering

Anonymous Author(s)
Affiliation
Address
email

Abstract

This paper introduces the first large-scale deep learning-based surrogate model for high-dimensional regression tasks in real-world mechanical engineering contexts. The model, comprising 43 million parameters, is trained on a custom in-house dataset, containing 2.8 billion data points from 31 million samples that are generated entirely through easy-to-evaluate, physics-based simulations. Each sample consists of 26 scalar features and 64 scalar targets. This large-scale synthetic dataset enables the training of deep neural networks over exhaustive and realistic mechanical design spaces. It exhibits complex statistical characteristics, including zero inflation, mutually exclusive features, strong multicollinearity, and a mix of real- and integer-valued data. Despite the scale and complexity of the dataset, the model is trained using entry-level consumer-grade graphics cards, thereby demonstrating the practical viability of deep learning for regression tasks in mechanical engineering applications.

1 Introduction

Deep neural networks (DNNs) have achieved impressive results in classification tasks in natural language processing, computer vision, and speech recognition. However, their adoption for regression tasks, despite being equally fundamental and widely used in science and engineering, has been more limited. In many fields, classical methods such as support vector machines, Gaussian processes or random forests remain the standard [1–4]. In engineering applications, physics-based computational tools like Finite Element Analysis continue to dominate [5], primarily due to the scarcity of large, high-quality datasets needed to train deep models effectively. The early development of neural networks illustrates the central role of data and computational resources in enabling deep learning. Although Multi-Layer Perceptrons (MLPs) were proposed in the 1960s, the practical use of neural networks remained limited for decades, not due to theoretical shortcomings, but because key requirements were lacking: large labeled datasets, sufficient computational power, and effective training techniques. These limitations were only overcome in the 2010s with the rise of big data [6], GPU architectures for parallel computing [7, 8], and improvements such as more efficient weight initialization [9], activation functions [10], and optimizers [11]. To this day, many regression problems in science and engineering still suffer from limited data availability, which restricts the effectiveness of deep learning approaches. While some studies attempt to apply deep learning in these settings, they often lack the amount of data required to fully exploit the capacity of such models [12–14]. Inspired by the success of pre-training in data-rich domains like natural language processing, we propose here a similar strategy for regression in Mechanical Engineering. To this end, we rely on a fast-to-evaluate physics-based model to efficiently generate large volumes of virtual data, thereby allowing for a thorough pre-training across a broad and representative input space, which would typically correspond to the

Submitted to 39th Conference on Neural Information Processing Systems (NeurIPS 2025). Do not distribute.

design space of a mechanical system. This enables us to construct and train on a dataset comprising 2.8 billion data points aggregated into 31 million samples, learning a high-dimensional mapping from 26 input features to 64 targets.

Even when neural networks are applied to regression tasks in engineering, they are often used with simplified architectures and training choices that limit their effectiveness. In many cases, the models consist of shallow networks with only a few layers and use outdated activation functions such as sigmoids [15–18]. This may be due to the limited data availability [19], which can make smaller and simple models appear sufficient [14, 20]. While these configurations were once common, substantial progress over the past decade has shown that deeper architectures with modern components, such as ReLU activations and better initialization schemes can significantly improve performance [10, 21]. Misconceptions also persist in several fields, such as the belief that increasing the network depth degrades accuracy [16], even though this has been refuted both theoretically and empirically [22–24]. Moreover, important approaches to improve and stabilize the training of the network such as GPU acceleration [16, 18] and applying regularization techniques like weight decay are often misused or underused [25], despite their widespread adoption [26]. These choices, while perhaps made for simplicity or due to the fact that the datasets are typically small, low in variability, and limited in feature richness, can prevent the training of DNNs that generalize well for engineering tasks such as design optimization.

Recent methods such as Physics-Informed Neural Networks (PINNs) [25, 27–30] and multi-fidelity modeling [31, 32] attempted to address the data scarcity in scientific and engineering domains. PINNs incorporate *a priori* physics knowledge into the training by penalizing deviations from governing equations, which can help guide learning when the amount of data is limited. However, choosing appropriate physical constraints and weighting them against the data-based loss remains difficult and can introduce bias [33]. In our case, physical knowledge is already embedded in the training data itself, since it is generated by a physics-based model. Multi-fidelity methods, on the other hand, aim to combine easy-to-evaluate low-fidelity with high-fidelity datasets, but in practice, they are often limited to small amounts of low-fidelity data due to computational cost or modeling assumptions. As a result, the pre-training remains limited, and a high reliance is placed on scarce, high-fidelity samples to attempt to achieve good generalization. To address this, we present a scalable framework for generating and training on billions of low-fidelity data points. This makes it possible to pre-train large-scale neural networks as surrogate models for regression tasks in a Mechanical Engineering context, with the expectation that such pre-training generalizes well and can be fine-tuned with minimal high-fidelity data. To our knowledge, this is the *very first practical implementation of DNN training at this scale for regression tasks in mechanical engineering.*

2 A representative example for high-dimensional regression in engineering

The deep learning framework presented in this work is general and designed to address complex, high-dimensional regression problems commonly found in engineering applications. To illustrate its capabilities, we focus on a representative use case from mechanical engineering: the prediction of excitation sources in gear systems. While gears are essential for transmitting motion and power efficiently, they also generate excitation sources that can lead to undesirable vibrations and noise. Our specific target is the Static Transmission Error (STE), the primary source of vibration and noise in gears. The STE quantifies the deviation between the actual and ideal position of the driven gear assuming rigid, perfectly conjugate geometry [34]. It stems from elastic deformations, which depend on the gear geometry and manufacturing errors, and is typically modeled under quasi-static conditions, which yield a scalar periodic function of the gear's rotation angle.

The aim of this paper is to learn a direct mapping from a fully parameterized gear geometry and operating conditions to the scalar function describing the STE. This predictive capability is critical for modeling system-level dynamics and designing quieter, more reliable mechanical systems. This application was chosen for its high industrial relevance, as gears are ubiquitous in many industries including the automotive, aerospace and industrial machinery sectors, and because it gives rise to a nonlinear, high-dimensional regression problem with complex statistical characteristics, including zero-inflated distributions and mutually-exclusive features. These characteristics make it both practically important and methodologically challenging, providing a strong test case for deep learning-based regression tasks.

The main contributions of the paper are summarized as follows:

- **Introduction** of a machine learning framework that uses large volumes of physics-based synthetic data to train a domain-specialized model capable of performing high-dimensional regression tasks.
- **Development** of a large-scale in-house dataset using physics-based simulations, making it possible to train surrogate models on data volumes comparable to those used in large language models, within a mechanical engineering context.
- **Release** of both the complete dataset and the trained surrogate model to support reproducibility and independent evaluation.

3 Dataset overview

The dataset consists of 26 input features defining the geometric and operating parameters of a gear pair, and 64-point output vectors representing the variation of STE across a full meshing cycle. To compute the STE, the cycle is discretized into 64 angular positions and a static equilibrium equation is solved at each position. The dataset covers a broad and representative set of physically valid gear configurations, based on an exhaustive exploration of the design space up to a chosen discretization, which eliminates the need for a separate pre-design phase. Physical validity is ensured through a set of geometric constraints applied to each configuration. Details regarding the methodology used for generating this dataset are thoroughly described in Appendix B.

3.1 Statistical properties

The dataset inputs consist of scalar parameters (e.g., `alpha_n_deg`) that define the studied mechanical system's geometry. While full physical descriptions and explanations of these parameters are provided in Appendix A, they are referenced by their dataset labels in the main text for conciseness. An in-depth statistical analysis of the dataset was carried out to identify its main characteristics and guide the design of the neural network-based surrogate model architecture. This section presents key findings from this analysis, illustrating important aspects of the data distribution that influenced our approach. The complete statistical details are reported in Appendix C.

Features. The analysis of the input features, shown in Fig. 1, reveals a statistically complex geometric parameter space. The features span vastly different scales tied to the physical quantitites they represent, including millimeters (e.g., `m_n`), micrometers (e.g., `C_beta1`), and newtons (`F`), as well as angular (e.g., `alpha_n_deg`) and non-dimensional units (e.g., `haP_et1`). The dataset includes zero-inflated features, often resulting from mutually exclusive parameters (e.g., `C_alphaTip1`, `tipReliefStartRadius1` with `C_alphaCrowning1`) or angular values frequently near zero (`beta_deg`). Integer-valued parameters (`z1`, `z2`) are also present. Inter-feature dependencies exist, with some parameters derived through linear or non-linear relationships, potentially introducing multicollinearity (e.g., `m_n` with `b`). Furthermore, many features, notably micro-geometry parameters (e.g., crowning and tip relief like `C_alphaCrowning1`, `C_alphaTip1`), display pronounced non-Gaussian characteristics. These distributions are frequently highly leptokurtic and exhibit significant positive skewness, marked by a high concentration of values near zero but possessing extended right tails.

Targets. The target variable is a 64-point vector of STE values, representing the mechanical system's static response. These values are physically bounded between 0 µm and approximately 600 µm. Across the dataset, the mean value of each target across all samples ranges from 45 µm and 53 µm, with standard deviations around 50 µm to 60 µm. The overall target distribution displays significant non-Gaussian properties, characterized by strong positive skewness (average ≈ 3.0) and high leptokurtosis (average ≈ 13.5), reflecting a notable prevalence of values near zero and infrequent extreme values.

3.2 Key statistical findings informing model design

The statistical analysis of the dataset highlights several critical properties that dictate the choice of our DNN architecture and training strategy. **Scale heterogeneity:** The wide variation, especially in input feature scales, necessitates input normalization. **Complex distributions:** While target and some feature variables exhibit positive skewness and leptokurtosis, deterministic preprocessing is

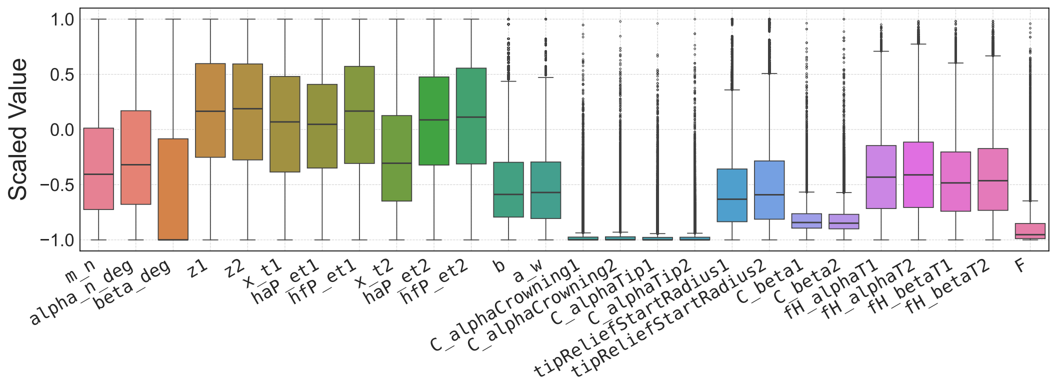


Figure 1: Box-plots representing the statistical distributions of the scaled input features computed with a 100,000-sample subset. The whiskers correspond to $\pm 1.5\times$ the interquartile range and outliers are shown as circle markers.

preferred over transformations (e.g., Yeo-Johnson [35]) to preserve the underlying relationships and avoid numerical artifacts in the physically-constrained output. **Dependencies and correlations:** The severe multicollinearity among input features and strong target-to-target correlations suggest that the dimensionality can be reduced. Given the non-linear relationship between the geometric parameters and the target STE values, autoencoders (AE) are preferred over linear techniques. Furthermore, the complex non-Gaussian distributions of both features and targets, coupled with the depth of the network, introduce the risk of gradient explosion during training. Therefore, batch normalization layers [36] are employed throughout the DNN to stabilize the gradient flow.

4 Experiments

This section presents the complete experimental procedure, including dimensionality reduction, DNN architecture optimization, and performance evaluation of the model components. The rationale behind each design choice is based on the statistical characteristics of the dataset and the practical constraints imposed by the available computational resources. The overall objective is to develop a surrogate model that balances complexity and predictive accuracy while remaining suitable for fast inference for typical engineering tasks such as design optimization.

In all the experiments, certain architectural and training components remain exactly the same. These shared characteristics include the selection of a standard neural network architecture, optimization methods, and training strategies. The architecture is based on a sequential arrangement of linear layers, batch normalization, and PReLU activation functions, with Kaiming Normal initialization [21] for weights and zero initialization for biases implemented in PyTorch. For optimization, the AdamW optimizer is used [11], and the OneCycleLR scheduler is employed for adaptive learning rate adjustment during training [37]. Additionally, gradient clipping by a max norm of 1.0 is applied to stabilize training [38]. Hyperparameter optimization is performed using Optuna with the Tree-structured Parzen Estimator (TPE) sampler [39], while model pruning is managed with the Median Pruner. These elements, detailed in Table 1, are provided for reproducibility purposes.

4.1 Implementation guidance and strategy

Hardware and computational constraints. A budget of USD 1,000 was allocated for GPU acquisition, resulting in the use of two Nvidia RTX 4060 Ti 16 GB GPUs, each delivering a theoretical peak performance of 22.06 TFLOPS in single precision. These GPUs are installed alongside an AMD Ryzen 9 5950X 16-core processor and 2 x 32 GB of DDR4 3200 MHz DIMM RAM. Although both GPUs are used in parallel, their memory cannot be combined, effectively limiting the usable VRAM to 16 GB per model. This constraint restricts the architectural complexity of the networks that can be explored, both in depth and in width, as large models quickly exceed the available memory. In addition, with only 22 TFLOPS of compute per GPU, performing extensive hyperparameter searches with models with more than 50 to 60 million parameters can quickly become intractable.

Table 1: Shared architectural and training components across all hyperparameter optimization experiments.

Component	Configuration
Number of Epochs	30
Training Precision	Single-precision format (float32)
Dataset splitting seed	42 (70% train, 15% validation and 15% test)
Layer Structure	Linear $\rightarrow$ BatchNorm $\rightarrow$ PReLU
PReLU Initialization	$\alpha = 0.25$
Weight Initialization	Kaiming Normal (nonlinearity: leaky ReLU)
Bias Initialization	Zeros
Optimizer	AdamW, $\beta_1 = 0.9$, $\beta_2 = 0.99$
Scheduler	OneCycleLR (cosine), 30% warm-up
Gradient Clipping	Max norm = 1.0

These limitations demand a careful balance between model size and predictive performance, with the objective of achieving fast and reliable inference suitable for real-time gear optimization. As a consequence, these hardware constraints motivated the reduction of input and output dimensionality prior to attempting the main neural network-based mapping.

Activation function and scaling strategy. The dataset is scaled using the `MinMaxScaler` from the scikit-learn Python library, with the range $[-1, 1]$, in line with the use of the PReLU activation function. This scaling choice avoids the dying ReLU problem [40], as PReLU activations mitigate the issue of dead units by allowing the slope of the negative part of the function to be learned. While this approach does not provide the smoothest loss curve compared to alternatives like GeLU, Swish, or Mish [41–43], it keeps the computation costs lower. GeLU, Swish, and Mish are based on trigonometric functions, which increase the computational expense.

Rationale behind using DNNs. To establish the necessity of non-linear modeling approaches, a Partial Least Squares Regression (PLSR) is implemented as a linear baseline. Using 19 components, the PLSR model achieved a Mean Adjusted R^2 Score of 0.3547 on the full test set (i.e., 15% of the dataset). This indicates that approximately two-thirds of the variance in the target variable could not be explained by a linear mapping of the input features. This emphasized the need for machine learning techniques capable of modeling complex non-linear relationships, leading to the selection of DNNs for mapping and autoencoders for unsupervised data pre-processing and dimensionality reduction, thanks to their representational strength and GPU-accelerated processing capabilities.

PCA for dimensionality reduction guidance. To guide the design of a lower-dimensional latent space for subsequent AEs training, Principal Component Analysis (PCA) is performed on both scaled input features and target variables. For the input features, 15 principal components were required to explain 95.0% of the variance, increasing to 19 for 99.0% and 24 for 99.9%. In contrast, the target variables demonstrated higher compressibility, with 3, 6, and 12 components capturing 95.0%, 99.0%, and 99.9% of the variance, respectively. This higher compressibility in the output is attributed to its time-series nature. These PCA results provided an initial empirical range for the latent dimensions of autoencoders: 12-20 for inputs and 2-12 for outputs.

4.2 Unsupervised data preprocessing

This subsection presents the use of autoencoders as an unsupervised preprocessing step to compress both the input and output spaces into lower-dimensional latent representations. The objective is to reduce computational cost while reshaping the data into compact, structured forms that facilitate efficient and effective subsequent mapping. This approach is particularly valuable given the highly complex and heterogeneous distributions within the dataset.

Autoencoder hyperparameter optimization. To determine the most effective model configurations for both input and target AEs, a series of hyperparameter searches were conducted using the Optuna framework. For both AE type, three distinct Optuna studies were performed, with each study dedicated to evaluating a specific loss function (MSE, MAE, Huber loss). Within each study, the searches explored a range of architectural parameters, such as network depth, layer widths, and dropout rates. All evaluated architectures incorporated internal batch normalization layers to promote training stability. Table 2 summarizes the best-performing feature and target AEs identified through these six Optuna studies. The entire optimization spanned 320 hours, with the input and output studies conducted independently on an Nvidia RTX 4060 Ti 16 GB GPU (160 hours each).

Table 2: Optimal hyperparameters and metrics for input and target autoencoders.

Hyperparameter Category	Feature AE	Target AE	Description / Sampling Range
Optuna Study Configuration			
Objective Metric	Min Reconstruction Loss		On the Complete Dataset
Total Searches	3	3	One search by loss type
Total Trials	100	100	Trials per search
Optuna Seed	42	42	For reproducibility concerns
Pruning	(15/15/1)	(15/15/1)	(Startup/Warmup/Interval)
Training Hyperparameters			
Batch Size	15,087	5,444	Log-int $[2^{10},\ 2^{14}]$
Max Learning Rate	7.4×10^{-3}	3.2×10^{-3}	Log-float $[10^{-4}, 3 \times 10^{-2}]$
Weight Decay	2.2×10^{-3}	3.0×10^{-3}	Log-float $[10^{-6},\ 10^{-2}]$
Loss Function	MSE	Huber	MSE, MAE, Huber
Best Huber δ	N/A	1.545	Log-float $[0.5,\ 2.0]$
Use tanh Output	False	True	True, False
Architectural Hyperparameters			
Original Dimension	26	64	Prior to compression
Bottleneck Dimension	15	5	Int $[12,\ 20]$ / Int $[2,\ 12]$
Bottleneck Dropout	0.0	0.0	Float $[0.0,\ 0.2]$
Hidden Layers	1	2	Int $[1,\ 5]$, before bottleneck
Units per HL	16	(53/38)	For encoder part
Dropout per HL	0.0	(0.0/0.0)	Float $[0.0,\ 0.5]$
Reconstruction Metrics			
R^2 Score	0.54266	0.99884	Variance Weighted
Adjusted R^2 Score	0.78389	0.99882	Uniform Average

Input compression discussion. Although multicollinearity analysis suggested an opportunity for input feature dimensionality reduction, autoencoders optimized via TPE struggled to effectively compress them. The best-performing autoencoder for reconstructing these features, despite its potential for non-linear mapping, explained only 78% of their variance (uniform R^2) and just 54% when weighted by feature variance. This performance was significantly less than the 95% of variance captured by 15 principal components derived from the same set of features. The fact that the optimizer selected a single-hidden-layer autoencoder as the optimal architecture further implies that deeper, non-linear structures provided no noticeable benefit for reconstruction accuracy. This observation aligns with theoretical insights indicating that single-layer autoencoders essentially learn PCA-like representations [44]. Considering these findings, and to avoid prematurely discarding potential non-linear relationships, dimensionality reduction through PCA was not applied. Instead, the analysis proceeded by utilizing the full set of min-max scaled input features.

Output compression discussion. For the output targets, autoencoder-based dimensionality reduction proved suitable, given their inherent sequential dependencies characteristic of time series data. An AE with a 5-dimensional bottleneck achieved an R^2 value of approximately 0.9988 for both uniform and variance-weighted metrics. This result indicates more effective compression compared to PCA, which needed 6 components to explain 99% of the variance and 12 components for 99.9%. Such performance shows the AE's ability to capture both linear and non-linear characteristics of the target signals. Notably, the TPE optimizer selected a tanh activation function for the AE's output layer, a choice that constrains predictions to the [-1, 1] range of the scaled data, thereby preventing potential artifacts during inverse transformation. This 5-dimensional encoding will therefore be employed for the targets in the subsequent mapping model to enhance computational efficiency.

4.3 Dimensionality-reduced feature-target mapping

Rationale behind the MLP architecture. Because the output signals are periodic by nature, there is no need for a model architecture capable of capturing long-term temporal dependencies, such as Long Short-Term Memory (LSTM) networks or Transformers. A standard Multilayer Perceptron (MLP) is therefore selected as the most suitable architecture for this regression task.

Table 3: Optimal hyperparameters and configuration for the MLP mapper identified via Optuna.

Hyperparameter Category	Value	Description / Sampling Range
Optuna Study Configuration		
Objective Metric	Min Huber Loss	Val dataset ($\delta = 1.0$)
Total Trials	75	52 completed, 23 pruned
Optuna Seed	25	For reproducibility concerns
Pruning	(20/12/1)	(Startup/Warmup/Interval)
Training Hyperparameters		
Batch Size	8,252	Log-int $[2^{11}, 2^{15}]$
Max Learning Rate	1.181×10^{-3}	Log-float $[10^{-5}, 5 \times 10^{-3}]$
Weight Decay	2.881×10^{-6}	Log-float $[10^{-6}, 10^{-2}]$
Loss Function	Huber	Fixed ($\delta = 1.0$)
Architectural Hyperparameters		
Input Dimension	26	Scaled Features
Output Dimension	5	Encoded Targets
Number of Hidden Layers	7	Int $[3, 8]$
Input Layer Dropout	0.113	Float $[0.0, 0.5]$
Neurons per Hidden Layer	See below	Log-int $[64, 4096]$
Dropout per Hidden Layer	See below	Float $[0.0, 0.5]$
Per-Layer Configuration (Units, Dropout)		
Hidden Layer 1	2,453	0.009
Hidden Layer 2	1,853	0.185
Hidden Layer 3	3,821	0.134
Hidden Layer 4	3,534	0.092
Hidden Layer 5	2,739	0.407
Hidden Layer 6	2,899	0.078
Hidden Layer 7	87	0.334
Validation Metrics		
R^2 Score	0.9978	Variance Weighted
Adjusted R^2 Score	0.9975	Uniform Average

MLP hyperparameter optimization. Having established the suitability of an MLP architecture, an extensive hyperparameter optimization process was undertaken using the Optuna framework to identify the most effective configuration for mapping the scaled input features to the compressed target representations. A Smooth L1 loss function (equivalent to Huber loss with $\delta = 1.0$) was employed for this regression task. This choice was motivated by its robustness to outliers, as it behaves like a L1 loss for errors larger than 1.0, reducing the strong penalization of large errors typical of MSE, while retaining smooth, L2-like behavior for smaller errors. The Optuna search explored a wide range of architectural parameters, including the number of hidden layers, neurons per layer, dropout rates, batch size, and learning rate. The detailed optimal hyperparameters and configuration details for the best-performing MLP mapper identified through this process are presented in Table 3.The hyperparameter search was conducted over a total of 272 hours on a single RTX 4060 Ti 16 GB.

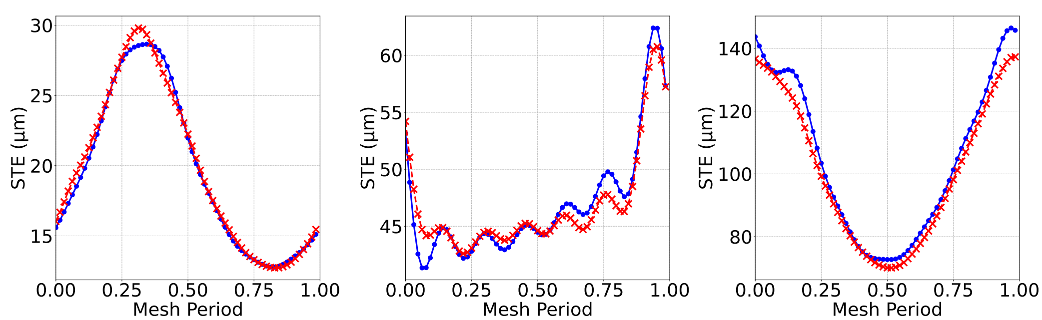
MLP mapper performance evaluation. Evaluation of the MLP mapper demonstrates its superior performance in mapping the AE's low-dimensional latent space to the 64-point STE profile. Achieving a Mean Adjusted R^2 of 0.9975, the MLP substantially outperforms the previous linear PLSR model, which scored 0.3547 on the same task. This confirms that the MLP successfully models the nonlinear relationship between the raw scaled geometric input parameters and the encoded output targets.

4.4 Surrogate model performance evaluation

Performance metrics. Evaluation of the final model was conducted on a test dataset comprising 4.66 million samples, with detailed performance metrics presented in Table 4. The model, characterized by 43.12 million trainable parameters distributed across 9 effective hidden layers (7 MLP, 2 Decoder), demonstrates excellent predictive accuracy, achieving an Adjusted R^2 of 0.9978. It also exhibits low relative errors, with a median absolute error of 1.70% and the 99th percentile error contained within 19.15% (both relative to the mean absolute value). Figure 2 shows the STE predictions corresponding to the 10th percentile, median and 95th percentile of the absolute prediction error. One can see a very good agreement between the predicted are reference targets in all three cases, which confirms the accuracy of the neural network. The model achieves an inference throughput of 95,095 samples/sec which allows for the prediction the STE for the entire 4.66 million-sample test set in approximately 49 seconds. This is several orders of magnitude faster than classical CPU-based computations, which would require an estimated 4,575,000 seconds (approximately 53 days) on an AMD Ryzen 9 5950X 16-core CPU. This represents a speedup factor of nearly 100,000 and makes it feasible to explore vast design spaces for design optimization tasks.

Table 4: Final model inference results on the full test dataset.

Metric Category	Value
Computational Performance (Single GPU)	
Total Pipeline Inference Time	49.02 sec
Throughput	95,095 samples/sec
Predictive Accuracy	
Adjusted R^2 Score (Uniform Avg)	0.9978
Relative Error Metrics	
Median Abs. Error as % of Mean Abs. Value	1.70%
MAE as % of Mean Abs. Value	2.96%
P95 MAE as % of Mean Abs. Value	9.83%
P99 MAE as % of Mean Abs. Value	19.15%
Normalized RMSE as % of Std Dev.	4.67%



(a) (b) (c)

Figure 2: STE predictions over a full meshing cycle corresponding to the 10th percentile (a) , median (b) and 95th percentile (c) of the absolute prediction error. The reference and predicted STE are shown as solid blue lines with circle marker (—●—) and solid red lines with cross markers (—×—), respectively. Note the different y-axis scales.

Feature importance and physical validation. To validate that the model has learned physically meaningful relationships, we assessed feature importance using permutation importance, measuring the drop in R^2 score upon shuffling each feature. The results, shown in Fig. 3, strongly align with the known physical principles governing the STE, with the applied load (`F`) being the most influential feature, causing an R^2 drop of over 0.8 when permuted. This is coherent with physical principles, where the applied load has a first-order effect on the static deflection of mechanical systems.

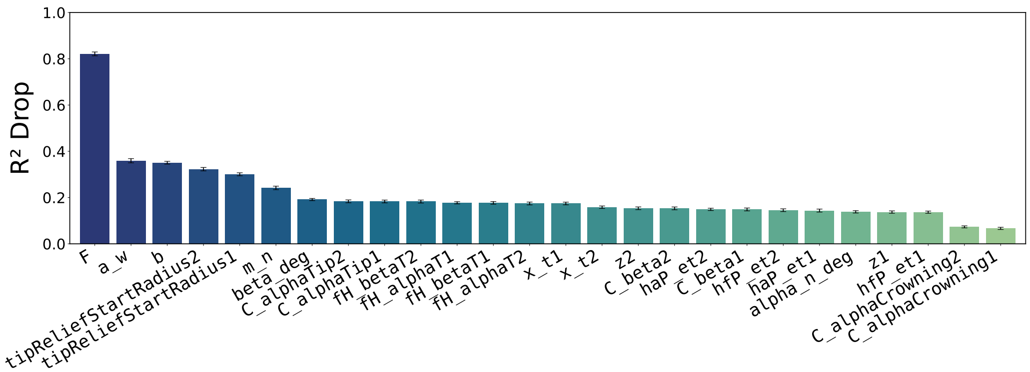


Figure 3: Permutation importance of input features, computed on a 20,000-sample test subset with 50 permutation repeats.

Parameters governing the stiffness of the gear, such as the face width (`b`) and operating center distance (`a_w`), are the next most influential, showing significant R^2 drops equal to approximately 0.35. The parameters controlling the STE profile shape and localized contact behavior, such as tip relief start radii (`tipReliefStartRadius1`, `tipReliefStartRadius2`), also demonstrate high importance (R^2 drops around 0.3). This ranking of feature importance, mirroring the hierarchy of physical effects on STE, confirms that the trained model has successfully captured the underlying physical principles of the system.

5 Conclusion

This paper presents the first neural network-based surrogate model trained on a dataset of this scale in engineering: 31 million samples and 2.8 billion data points. While fully data-driven modeling is gaining momentum in science and engineering, its practical use remains limited, largely due to the scarcity of high-quality data. We show that combining deep learning with low-fidelity, physics-based synthetic data offers a viable path forward. These data are fast to generate and, by construction, incorporate fundamental aspects of the underlying physics. They make it possible to create large and diverse datasets for training domain-specialized surrogates that capture key physical behaviors and generalize well. The surrogate model presented in this work is highly accurate with an adjusted R^2 equal to 0.9978. These results, obtained despite architectural and training constraints due to hardware limitations, give confidence in the broad applicability of the approach. This work provides a foundation for training accurate models that can be fine-tuned using small amounts of high-fidelity experimental or simulation data. It opens a path toward the efficient inference and design optimization of mechanical systems, thereby bringing practical industrial applications within reach.

References

[1] A. Statnikov, L. Wang, and C.F. Aliferis. A comprehensive comparison of random forests and support vector machines for microarray-based cancer classification. *BMC Bioinformatics*, 9:319, 2008.

[2] C. Hultquist, G. Chen, and K. Zhao. A comparison of Gaussian process regression, random forests and support vector regression for burn severity assessment in diseased forests. *Remote Sensing Letters*, 5: 723–732, 2014.

[3] G. Teles, J. J. P. C. Rodrigues, R. A. L. Rabêlo, and S. A. Kozlov. Comparative study of support vector machines and random forests machine learning algorithms on credit operation. *Software: Practice and Experience*, 51:2492–2500, 2021.

[4] M. R. Ebers, K. M. Steele, and J. N. Kutz. Discrepancy Modeling Framework: Learning Missing Physics, Modeling Systematic Residuals, and Disambiguating between Deterministic and Random Effects. *SIAM Journal on Applied Dynamical Systems*, 23(1):440–469, 2024.

[5] W. K. Liu, S. Li, and H. S. Park. Eighty Years of the Finite Element Method: Birth, Evolution, and Future. *Archives of Computational Methods in Engineering*, 29:4431–4453, 2022.

[6] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, Miami, FL, USA, 2009.

[7] R. Raina, A. Madhavan, and A. Y. Ng. Large-scale deep unsupervised learning using graphics processors. In *Proceedings of the 26th Annual International Conference on Machine Learning*, Montreal, Quebec, Canada, 2009.

[8] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, volume 25, 2012.

[9] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pages 249–256, 2010.

[10] A. L. Maas, A. Y. Hannun, and A. Y. Ng. Rectifier Nonlinearities Improve Neural Network Acoustic Models. In *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, Atlanta, Georgia, USA, 2013.

[11] I. Loshchilov and F. Hutter. Decoupled Weight Decay Regularization. In *International Conference on Learning Representations (ICLR)*, 2019.

[12] Z. Guo, M. Cucuringu, and A. Y. Shestopaloff. Generalized factor neural network model for high-dimensional regression. *arXiv preprint*, 2025.

[13] K. A. Wang, M.E. Levine, J. Shi, and E. B. Fox. Learning absorption rates in glucose-insulin dynamics from meal covariates. In *NeurIPS 2022 Workshop on Learning from Time Series for Health*, 2022.

[14] N. B. Erichson, L. Mathelin, Z. Yao, S. L. Brunton, M. W. Mahoney, and J. N. Kutz. Shallow neural networks for fluid flow reconstruction with limited sensors. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 2020.

[15] J. N. Heidenreich and D. Mohr. Extended minimal state cells (EMSC): Self-consistent recurrent neural networks for rate- and temperature dependent plasticity. *International Journal of Plasticity*, 188:104305, 2025.

[16] E. Sakaridis, C. Kalligeros, C. Papalexis, G. Kostopoulos, and V. Spitas. Symmetry preserving neural network models for spur gear static transmission error curves. *Mechanism and Machine Theory*, 187: 105 369, 2023.

[17] B. Z. Cunha, C. Droz, A. Zine, M. Ichchou, and S. Foulard. Neural Network-based Surrogates of Gear Whine Noise for Uncertainty Propagation. In *7th European Conference on Structural Control: Book of Abstracts and Selected Papers*, Warsaw, Poland, 2022.

[18] M. Willecke, J. Brimmers, and C. Brecher. Surrogate model based prediction of transmission error characteristics based on generalized topography deviations. *Forschung im Ingenieurwesen*, 87:431–440, 2023.

[19] B. Z. Cunha, C. Droz, A. Zine, S. Foulard, and M. Ichchou. A review of machine learning methods applied to structural dynamics and vibroacoustic. *Mechanical Systems and Signal Processing*, 200:110535, 2023.

[20] S. Chauhan, L. Vig, M. De Filippo De Grazia, M. Corbetta, S. Ahmad, and M. Zorzi. A Comparison of Shallow and Deep Learning Methods for Predicting Cognitive Performance of Stroke Patients From MRI Lesion Images. *Frontiers in Neuroinformatics*, 13:53, 2019.

[21] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Santiago, Chile, 2015.

[22] K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2:359–366, 1989.

[23] M. Kohler and S. Langer. On the rate of convergence of fully connected deep neural network regression estimates. *The Annals of Statistics*, 49(4):2231–2249, 2021.

[24] P. Nakkiran, G. Kaplun, Y. Bansal, T. Yang, B. Barak, and I. Sutskever. Deep Double Descent: Where Bigger Models and More Data Hurt. *arXiv preprint*, 2019.

[25] X. Jian, K. Bacsa, G. Duthé, and E. Chatzi. Modal Decomposition and Identification for a Population of Structures Using Physics-Informed Graph Neural Networks and Transformers. *arXiv preprint*, 2025.

[26] F. D'Angelo, M. Andriushchenko, A. Varre, and N. Flammarion. Why Do We Need Weight Decay in Modern Deep Learning? In *Advances in Neural Information Processing Systems 37 (NeurIPS 2024)*, 2024.

[27] Z. Lai, C. Mylonas, S. Nagarajaiah, and E. Chatzi. Structural identification with physics-informed neural ordinary differential equations. *Journal of Sound and Vibration*, 508:116196, 2021.

[28] J. N. Fuhg and N. Bouklas. On physics-informed data-driven isotropic and anisotropic constitutive models through probabilistic machine learning and space-filling sampling. *Computer Methods in Applied Mechanics and Engineering*, 394:114915, 2022.

[29] A. Aygun, R. Maulik, and A. Karakus. Physics-informed neural networks for mesh deformation with exact boundary enforcement. *Engineering Applications of Artificial Intelligence*, 125:106660, 2023.

[30] T. G. Grossmann, U. J. Komorowska, J. Latz, and C.-B. Schönlieb. Can physics-informed neural networks beat the finite element method? *IMA Journal of Applied Mathematics*, 89:143–174, 2024.

[31] P. Conti, M. Guo, A. Manzoni, and J. S. Hesthaven. Multi-fidelity surrogate modeling using long short-term memory networks. *Computer Methods in Applied Mechanics and Engineering*, 404:115811, 2023.

[32] P. Conti, M. Guo, A. Manzoni, A. Frangi, S. L. Brunton, and J. N. Kutz. Multi-fidelity reduced-order surrogate modelling. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 480, 2024.

[33] N. McGreivy and A. Hakim. Weak baselines and reporting biases lead to overoptimism in machine learning for fluid-related partial differential equations. *Nature Machine Intelligence*, 6:1256–1269, 2024.

[34] D. B. Welbourn. Fundamental knowledge of gear noise: A survey. In *Noise and Vibrations of Engines and Transmissions*, Cranfield, England, 1979.

[35] I.-K. Yeo and R. A. Johnson. A New Family of Power Transformations to Improve Normality or Symmetry. *Biometrika*, 87:954–959, 2000.

[36] S. Ioffe and C. Szegedy. Batch normalization: accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 448–456, 2015.

[37] L. N. Smith and N. Topin. Super-convergence: Very fast training of neural networks using large learning rates. *arXiv preprint*, 2017.

[38] J. Zhang, T. He, S. Sra, and A. Jadbabaie. Why Gradient Clipping Accelerates Training: A Theoretical Justification for Adaptivity. In *International Conference on Learning Representations (ICLR)*, Cambridge, MA, USA, 2020.

[39] S. Watanabe. Tree-Structured Parzen Estimator: Understanding Its Algorithm Components and Their Roles for Better Empirical Performance. *arXiv preprint*, 2023.

[40] L. Lu, Y. Shin, Y. Su, and G. E. Karniadakis. Dying ReLU and Initialization: Theory and Numerical Examples. *arXiv preprint*, 2019.

[41] D. Hendrycks and K. Gimpel. Gaussian Error Linear Units (GELUs). *arXiv preprint*, 2016.

[42] P. Ramachandran, B. Zoph, and Q. V. Le. Searching for Activation Functions. *arXiv preprint*, 2017.

[43] D. Misra. Mish: A Self Regularized Non-Monotonic Activation Function. *arXiv preprint*, 2019.

[44] H. Bourlard and Y. Kamp. Auto-association by multilayer perceptrons and singular value decomposition. *Biological Cybernetics*, 59:291–294, 1988.

[45] M. Szwarcman. *Éléments de machines*. Technique et documentation-Lavoisier, Paris, 1983.

[46] A. L. Rosado, F. P. Muñoz, and R. A. Fernández. An Analytic Expression for the Inverse Involute. *Mathematical Problems in Engineering*, 2019, 2019.

[47] K. L. Johnson. *Contact Mechanics*. Cambridge University Press, Cambridge, 1985.

[48] P. Garambois, J. Perret-Liaudet, and E. Rigaud. NVH robust optimization of gear macro and microgeometries using an efficient tooth contact model. *Mechanism and Machine Theory*, 117:78–95, 2017.

[49] Y. Benaïcha, J. Perret-Liaudet, J. D. Beley, E. Rigaud, and F. Thouverez. On a flexible multibody modelling approach using FE-based contact formulation for describing gear transmission error. *Mechanism and Machine Theory*, 167:104505, 2022.

[50] M. Benatar, M. Handschuh, A. Kahraman, and D. Talbot. Static and Dynamic Transmission Error Measurements of Helical Gear Pairs With Various Tooth Modifications. *Journal of Mechanical Design*, 141:103301, 2019.

A Nomenclature of input features

Table 5 associates each input feature with its physical counterpart. The input variables (e.g., `m_n`) are presented in the same order as expected by the input of the surrogate model. In all subsequent appendices, the designations presented in Table 5 will be used in place of the variable names.

Table 5 is organized as follows:

- **11 minimal macro-geometric parameters:** Minimal set of macro-geometric parameters that cannot be derived from others which can not be deduced from others.
- **1 deduced macro-geometric parameter:** The only macro-geometric parameter that can be derived from the normal module m_n.
- **2 operational conditions:** Defined by the chosen center distance between the two gears and the load to be transmitted.
- **8 tooth flank modifications:** Also called "micro-geometry deviations", "micro-geometry modifications", "micro-geometry corrections", "tooth flank corrections" or simply "micro-geometries", these represent intentional deviations from the theoretical tooth profile. Note that tip relief ($C_{\alpha a}$, $r_{c\ \alpha a}$) and profile crowing C_α are mutually exclusive modifications.
- **4 tooth flank tolerances:** These correspond to manufacturing errors.

Table 5: Nomenclature of input features

Name	Input Variable	Designation	Unit
Minimal macro-geometric parameters (according to ISO 53, ISO 1122 and ISO 6336-1)			
Normal module	`m_n`	m_n	mm
Normal pressure angle	`alpha_n_deg`	α_n	deg
Helix angle	`beta_deg`	β	deg
Number of teeth of gear 1	`z_1`	z_1	-
Number of teeth of gear 2	`z_2`	z_2	-
Profile shift coefficient of gear 1	`x_t1`	x_1	-
Addendum coefficient of gear 1	`haP_et1`	$h_{aP}{}_1^*$	-
Dedendum coefficient of gear 1	`hfP_et1`	$h_{fP}{}_1^*$	-
Profile shift coefficient of gear 2	`x_t2`	x_2	-
Addendum coefficient of gear 2	`haP_et2`	$h_{aP}{}_2^*$	-
Dedendum coefficient of gear 2	`hfP_et2`	$h_{fP}{}_2^*$	-
Deduced macro-geometric parameter (according to ISO 1122 and ISO 6336-1)			
Facewidth	`b`	b	mm
First operational condition (according to ISO 1122 and ISO 6336-1)			
Operating center distance	`a_w`	a_{t_w}	mm
Tooth flank modifications (according to ISO 21771:2024)			
Profile crowning (barreling) of gear 1	`C_alphaCrowning1`	C_{α_1}	µm
Profile crowning (barreling) of gear 2	`C_alphaCrowning2`	C_{α_2}	µm
Tip relief of gear 1	`C_alphaTip1`	$C_{\alpha a_1}$	µm
Tip relief of gear 2	`C_alphaTip2`	$C_{\alpha a_2}$	µm
Tip relief start radius of gear 1	`tipReliefStartRadius1`	$r_{c\ \alpha a_1}$	mm
Tip relief start radius of gear 2	`tipReliefStartRadius2`	$r_{c\ \alpha a_2}$	mm
Flank line (helix) crowning of gear 1	`C_beta1`	C_{β_1}	µm
Flank line (helix) crowning of gear 2	`C_beta2`	C_{β_2}	µm
Tooth flank tolerances for an ISO quality class 6 (according to ISO 1328-1:2013)			
Profile slope tolerance of gear 1	`fH_alphaT1`	$f_{H\alpha T_1}$	µm
Profile slope tolerance of gear 2	`fH_alphaT2`	$f_{H\alpha T_2}$	µm
Helix form tolerance of gear 1	`fH_betaT1`	$f_{H\beta T_1}$	µm
Helix form tolerance of gear 2	`fH_betaT2`	$f_{H\beta T_2}$	µm
Second operational condition (according to ISO 6336-1)			
Transmitted load	`F`	F	N

B Dataset generation

B.1 Gear design space

The gear design space, from which the dataset is generated, is constructed by defining key gear macro-geometric and micro-geometric parameters. This space encompasses the entire set of physically valid external cylindrical gears within a specified range, defined by 11 minimal parameters essential for describing gear macro-geometry (i.e., parameters that cannot be deduced from others). To ensure an as exhaustive as possible representation of this space, it is populated using Latin Hypercube Sampling (LHS). The physical validity of the sampled gears is guaranteed through the application of macro-geometric constraints, derived from established gear theory principles extensively detailed and proved in [45]. Micro-geometric deviations, by contrast, are treated separately. These deviations represent variations from the nominal tooth profile. They may be intentionally introduced (e.g., reduce the fluctuation of the STE during the meshing cycle), or they may arise unintentionally from manufacturing or assembly errors. The complete list of considered parameters can be found in Appendix A.

B.2 Fundamentals of gear theory

Gears are essential mechanical components. They are among the most commonly used, most robust, and most durable systems to transmit motion and power. By definition, a gear is a basic mechanism made up of two toothed wheels that rotate around axes with a fixed relative position, where one drives the other through the successive engagement of their teeth.

In this paper, two types of gears are considered:

- Spur gears, which are the simplest and most economical. They are used to transmit motion and power between two parallel shafts. The teeth of both gears are parallel to the rotation axis of the shafts.
- Helical gears, in which the teeth are inclined relatively to the axis of rotation of the shafts. They are also used to transmit motion and power between two parallel shafts. For the same size, they are more efficient than spur gears in transmitting power and torque. Due to the more gradual and continuous engagement of the teeth, they are also quieter.

B.2.1 Basic definitions

A gear contains an integer number of teeth z_i, each spaced at equal intervals corresponding to the normal pitch p_n. This leads to the following relation:

$$\pi\, d_i = p_n\, z_i \tag{1}$$

where d_i is the pitch diameter.

By introducing the normal module m_n, defined as:

$$m_n = \frac{p_n}{\pi} \tag{2}$$

The expression leads to:

$$d_i = m_n\, z_i \tag{3}$$

Thus, meshing is only possible if the normal module of each gear is equal. Note that the strength of the tooth depends on the selected normal module.

B.2.2 Gear kinematics

The most commonly used tooth profile in gears is the involute profile, as it is the only profile that guarantees a constant gear ratio regardless of the operating center distance a_w:

$$\frac{\omega_2}{\omega_1} = \frac{r_1}{r_2} = \frac{z_1}{z_2} \tag{4}$$

where ω_i is the angular speed of gear i and $r_i = d_i/2$ is the pitch radius.

The involute is generated by a point attached to a rigid line that rolls without slipping along the circumference of a fixed circle, known as the base circle, with radius r_b. It is the result of unwinding a wire that is wrapped around the base circle. It is defined in polar coordinates as:

$$\begin{cases} r_M = \dfrac{r_b}{\cos(\alpha_M)} \\ \\ \theta_M = \tan(\alpha_M) - \alpha_M \end{cases} \tag{5}$$

given that,

$$\text{inv}(\alpha_M) = \tan(\alpha_M) - \alpha_M \tag{6}$$

where, the point M is the one that generates the involute, r_M the radius at the point M, θ_M the angular position of the point M, α_M the pressure angle at the point M, which defines the direction of the force transmitted by the gear teeth, and increases as r_M increases. $\text{inv}(\alpha_M)$ represents the involute function. Consequently, the gear can be viewed as a line that rolls without slipping on two base circles, which serve as the starting points for the respective involute profiles of each gear. The base radius r_{b_i} is therefore expressed as:

$$r_{b_i} = r_i \cos(\alpha_n) \tag{7}$$

The path of contact begins at point A, where the tip circle of the driven gear intersects the line of action, and ends at point E, where the tip circle of the driving gear intersects the same line. The tip radius r_{a_i} of a gear is related to its pitch radius r_i and addendum h_{a_i} by:

$$r_{a_i} = r_i + h_{a_i} \tag{8}$$

Similarly, the root radius is defined as:

$$r_{f_i} = r_i - h_{f_i} \tag{9}$$

where h_{f_i} is the dedendum.

B.2.3 Static transmission error definition

To ensure continuous meshing, there must always be at least one pair of teeth in contact at any given time. Therefore, the base pitch p_{b_n} which is the distance between two consecutive teeth along the base circle, must be strictly less than the length of path of contact g_α, defined as the distance from the point A where the contact begins, and the point where it ends E. The base pitch p_{b_n} can be expressed from the normal pitch p_n and the normal pressure angle α_n:

$$p_{b_n} = p_n \cos(\alpha_n) \tag{10}$$

As a result, the number of tooth pairs in contact varies over time, giving rise to a periodic excitation which is periodic to the mesh period. This excitation is periodic because, in the absence of manufacturing defects, the base pitch p_{b_n} remains constant. Therefore, in reality, the gear ratio is not strictly constant, as previously assumed, but instead fluctuates around its mean value.

The variation in the number of teeth in contact leads to changes in the apparent stiffness of the gear mesh. Indeed, if the load is shared between two tooth pairs, each tooth supports only half the load compared to the case where a single pair carries the entire load. This variation results in fluctuating tooth deflection under load during the meshing process. Such deflection can be modeled as a relative displacement along the line of action, known as the static transmission error (STE), and defined as:

$$\delta(\theta_1) = r_{b_2}\,\theta_2 - r_{b_1}\,\theta_1 \tag{11}$$

where θ_1 and θ_2 are respectively the angular position of the driven and the driving gears, and $\delta(\theta_1)$ the static transmission error usually expressed in µm. The STE can thus be understood as the deviation between the theoretical position of the driven gear, assuming perfectly rigid bodies and ideal meshing, and its actual position under load.

The time-varying mesh stiffness is thereby computed from the STE as:

$$k(\theta_1) = \frac{\partial F}{\partial\, \delta(\theta_1)} \tag{12}$$

where F is the transmitted load and $k(\theta_1)$ the time-varying mesh stiffness. Accordingly, predicting mesh stiffness is unnecessary for the surrogate, as it is easily obtained through numerical differentiation, as is standard with physics-based models.

B.2.4 Gear cutting

During cutting, the normal pressure angle must not exceed a certain threshold value. When $\alpha_n = 0$ deg, the system is called a rack and pinion. Note that involute gears are typically manufactured using a rack-cutting tool. For gears, pressure angles typically range between 15 and 25 deg.

If the cutting rack is perpendicular to the axis of the pinion, the gear is referred to as a spur gear. Conversely, if an angle is introduced and the rack is offset from the perpendicular to the pinion axis, the teeth are cut in a helical shape, and the gear is then referred to as a helical gear. This angle is known as the helix angle β. Note that the term "helix angle" implicitly refers to the pitch helix. The base helix angle β_b is also defined as follows:

$$\beta_b = \tan^{-1}\left(\frac{r_{b_i}}{r_i\,\tan(\beta)}\right) \tag{13}$$

To geometrically describe a helical gear, the concept of the transverse plane is introduced, which is perpendicular to the axis of rotation of the pinion. It is denoted by the letter t. The rack plane, on the other hand, is referred to as the normal plane and is denoted by n. The transverse pressure angle α_t, the transverse module m_t, the transverse pitch p_t and the transverse base pitch p_{b_t} can be expressed as:

$$\alpha_t = \tan^{-1}\left(\frac{\tan(\alpha_n)}{\cos(\beta)}\right) \tag{14}$$

$$m_t = \frac{m_n}{\cos(\beta)} \tag{15}$$

$$p_t = \pi\, m_t \tag{16}$$

$$p_{b_t} = p_t\,\cos(\alpha_t) \tag{17}$$

During the cutting process, the cutting rack rolls without slipping on the blank tip cylinder (i.e., the cylinder prior to machining) of a given facewidth b. It is possible to offset the reference plane of the rack from the pitch cylinder of the gear being cut. Note that the reference plane of the rack is geometrically equivalent to the pitch cylinder of a gear. This offset is referred to as the profile shift. By convention, the profile shift is considered positive when the reference plane of the cutting rack is displaced outward from the pitch cylinder, and negative when it is displaced inward, penetrating into the pitch cylinder of the gear being cut. To quantify the profile shift, a dimensionless profile shift coefficient is introduced. This coefficient is denoted by x_{t_i}. A normal profile shift coefficient x_{n_i} can thus be derived, and expressed as:

$$x_{n_i} = \frac{x_{t_i}\,m_t}{m_n} \tag{18}$$

Note that the profile shift coefficient is commonly represented by x_i; however, a distinction is made in [45] between the classical profile shift coefficient and the normal profile shift coefficient, the latter being used for subsequent calculations of gear validity constraints.

After the gear is cut, the addendum of the tooth can be further reduced by an additional machining operation known as truncation. This process removes a portion of the tooth tip to modify the tooth height, and is quantified by the truncation coefficient k_i.

B.2.5 Induced gear parameters

Using the previously defined parameters, the following induced quantities can be calculated and will be employed for subsequent calculations of gear validity constraints. These relationships are well established in the literature [45] and do not constitute novel findings.

The tooth addendum and dedundum can be expressed as:

$$h_{a_i} = m_n\,\left(h_{aP}{}_i^* + x_{t_i} - k_i\right) \tag{19}$$

$$h_{f_i} = m_n\,\left(h_{fP}{}_i^* - x_{t_i}\right) \tag{20}$$

where ${h_{aP}}_i^*$ and ${h_{fP}}_i^*$ are respectively addendum and dedendum coefficients.

The axial pitch p_x is expressed as:

$$p_x = \frac{p_n}{\sin(\beta)} \tag{21}$$

For an helical gear, the lead p_z is defined as:

$$p_z = \frac{z_i \, \pi \, m_n}{\sin(\beta)} \tag{22}$$

The transverse pitch tooth thickness s_{t_i}, the length of the arc of the pitch circle lying between two profiles of a tooth, is formulated as:

$$s_{t_i} = \frac{\pi \, m_n}{2} + 2 \, x_{t_i} \, m_n \, \tan(\alpha_t) \tag{23}$$

Similarly, the transverse pitch spacewidth e_{t_i}, the length of the arc of the pitch circle lying between two teeth, is described as:

$$e_{t_i} = \frac{\pi \, m_n}{2} - 2 \, x_{t_i} \, m_n \, \tan(\alpha_t) \tag{24}$$

The angle at the tip end of the involute α_{a_i} is defined as:

$$\alpha_{a_i} = \cos^{-1}\left(\frac{r_{b_i}}{r_{a_i}}\right) \tag{25}$$

The transverse tip tooth thickness $s_{a_t 2}$ of the gear 2 is formulated as:

$$s_{a_t 2} = 2 \, r_{a_2} \, \frac{s_{t_1}}{2 \, r_2 + \text{inv}(\alpha_t) - \text{inv}(\alpha_{a_2})} \tag{26}$$

and conversely for the gear 1.

B.3 Macro-geometric constraints

For the sake of completeness and reproducibility reasons, we summarize the main macro-geometric constraints that allow the gear to be manufactured and assembled. The interested readers can find more detailed in this book chapter [45], all preceding Subsections of this Appendix provide the minimal background necessary to follow it.

The most challenging aspect is determining the minimum operating center distance, which involves the use of the inverse involute function (see Eq. 6 for the definition of the involute function). Since this function cannot be expressed mathematically in a direct way, it is generally evaluated using approximate formulas. **This issue is addressed using the procedure described in the Paragraph named "Valid operating center distances for a valid gear pair" which, to the best of the authors' knowledge, represents a novel contribution to the literature**. This approach further enables the identification of all valid operating center distances for each gear pair, allowing the construction of an exhaustive dataset that is not limited to the theoretical center distance a_{th} defined as:

$$a_{\text{th}} = \frac{m_t \, (z_1 + z_2)}{2} \tag{27}$$

Design space bounds

The design space bounds correspond to the limits of the 11 minimal macro-geometric parameters and the derived macro-geometric parameters described in Appendix A.

$$0.5 \text{ mm} \leq m_n \leq 12 \text{ mm} \tag{28}$$

$$15^\circ \leq \alpha_n \leq 25^\circ \tag{29}$$

$$0^\circ \leq \beta \leq 30^\circ \tag{30}$$

$$11 \leq (z_1,\, z_2) \leq 150 \tag{31}$$

$$0.5 \leq ({h_{aP}}_1^*,\, {h_{aP}}_2^*,\, {h_{fP}}_1^*,\, {h_{fP}}_2^*) \leq 2.0 \tag{32}$$

$$-1.0 \leq (x_{t_1},\, x_{t_2}) \leq 1.0 \tag{33}$$

$$5.0 \leq \frac{b}{m_n} \leq 15 \tag{34}$$

Trivial conditions

Since the macro-parameters are selected via LHS within the design space bounds, a set of simple preliminary conditions is applied to filter out invalid candidates. Then, they are passed through and sequentially filtered by each of the subsequent conditions presented in this subsection.

$$r_{a_1} > {r_f}_1 \wedge r_{a_2} > {r_f}_2 \tag{35}$$

$$r_{a_1} > r_1 \wedge r_{a_2} > r_2 \tag{36}$$

$$r_{a_1} > r_{b_1} \wedge r_{a_2} > r_{b_2} \tag{37}$$

$$r_1 > r_{b_1} \wedge r_2 > r_{b_2} \tag{38}$$

$$r_1 > {r_f}_1 \wedge r_2 > {r_f}_2 \tag{39}$$

Minimal transverse tooth thickness at the tip circle condition

$$s_{a_1} \geq 0.1\, m_t \wedge s_{a_2} \geq 0.1\, m_t \tag{40}$$

Minimal contact ratio condition

$$\varepsilon_\gamma = \varepsilon_\alpha + \varepsilon_\beta \geq 1.2 \text{ with } \varepsilon_\alpha \geq 1.0 \tag{41}$$

Minimal number of teeth condition

$$z_{\text{lim}} = \frac{2\cos(\beta)}{\sin^2(\alpha_t)} \tag{42}$$

No manufacturing interference conditions

$$\frac{z_1}{2}\sin^2(\alpha_t) - (1 - x_{t_1}) \geq 0 \tag{43}$$

$$\frac{z_2}{2}\sin^2(\alpha_t) - (1 - x_{t_2}) \geq 0 \tag{44}$$

Assembly conditions

$$\text{clearance}_1 = a_w - r_{a_1} - r_{f\,2} > 0 \tag{45}$$

$$\text{clearance}_2 = a_w - r_{a_2} - r_{f\,2} > 0 \tag{46}$$

$$\text{inv}(\alpha_{t_w}) \geq 2\left(\frac{x_{n_1} - x_{n_2}}{z_1 + z_2}\right)\tan(\alpha_n) + \text{inv}(\alpha_t) \tag{47}$$

No meshing interference conditions

$$d_{a_1} \leq z_1\, m_t\, \cos(\alpha_t)\sqrt{1 + \left(\frac{z_2}{z_1}\,(\tan(\alpha_{t_w}) - \tan(\alpha_t)) + \tan(\alpha_{t_w}) + \frac{4\,(h_{aP}{}_2^* - x_{t_2})}{z_1\,\sin(2\,\alpha_t)}\right)^2} \tag{48}$$

$$d_{a_2} \leq z_2\, m_t\, \cos(\alpha_t)\sqrt{1 + \left(\frac{z_1}{z_2}\,(\tan(\alpha_{t_w}) - \tan(\alpha_t)) + \tan(\alpha_{t_w}) + \frac{4\,(h_{aP}{}_1^* - x_{t_1})}{z_2\,\sin(2\,\alpha_t)}\right)^2} \tag{49}$$

Tooth root stress conditions

This work considers steel as the sole material. The yield strength at the tooth root $\sigma_{F\,\text{max}}$ is set to 500 MPa.

$$F_{1\,\text{max}} = \frac{\sigma_{F\,\text{max}}\, b\, \pi^2\, m_n\, \varepsilon_\alpha}{\cos(\alpha_{t_w})\left(2\sin(\alpha_{t_w})\,\varepsilon_\alpha\,\pi\; +\; 24\cos(\alpha_{t_w})\,(h_{aP}{}_1^* + h_{fP}{}_1^*)\right)} \tag{50}$$

$$F_{2\,\text{max}} = \frac{\sigma_{F\,\text{max}}\, b\, \pi^2\, m_n\, \varepsilon_\alpha}{\cos(\alpha_{t_w})\left(2\sin(\alpha_{t_w})\,\varepsilon_\alpha\,\pi\; +\; 24\cos(\alpha_{t_w})\,(h_{aP}{}_2^* + h_{fP}{}_2^*)\right)} \tag{51}$$

Contact pressure condition

The steel yield strength at contact pressure $\sigma_{H\,\text{max}}$ is set to 1200 MPa. Young's moduli E_1, E_2 and Poisson's ratios ν_1, ν_2 are taken respectively as 210 GPa and 0.29.

$$F_{H\,\text{max}} = \frac{\sigma_{H}{}^2_{\text{max}}\, b\, \varepsilon_\alpha\, \pi\, \left(\frac{1-\nu_1^2}{E_1} + \frac{1-\nu_2^2}{E_2}\right)\cos(\alpha_{t_w})}{\cos(\beta)\left(\frac{1}{T_1C} + \frac{1}{T_2C}\right)} \tag{52}$$

Valid operating center distances for a valid gear pair

Each physically valid gear pair admits a minimum operating center distance $a_{w\,\text{min}}$ imposed by the assembly conditions (Eqs. 45 and 46), and a maximum operating center distance $a_{w\,\text{max}}$ imposed by the minimal contact ratio condition (Eq. 47). To determine all the valid operating center distances a_w, several candidate values are evaluated within the range defined by the following bounds:

$$a_{w\,\text{min}} = r_{f\,1} + r_{f\,2} \tag{53}$$

$$a_{w\,\text{max}} = r_{a\,1} + r_{a\,2} \tag{54}$$

where $r_{f\,i}$ and $r_{a\,i}$ are the dedendum and addendum radii of the pinion and gear. Ten uniformly spaced values of a_w are sampled between $a_{w\,\text{min}}$ and $a_{w\,\text{max}}$, for each gear pair.

Candidate values that do not satisfy Eqs. 45 and 46 are directly discarded. For Eq. 47, the transverse operating pressure angle is computed from each generated center distance and passed through the involute function (Eq. 6) which is not invertible [46]. Given that the involute function is strictly increasing in the range of 15 to 25 deg, and that the transverse operating pressure angle increases with an increasing center distance, invalid values of a_w can be easily filtered out thanks to the inequality defined in Eq. 47. This procedure avoids the need for iterative optimization algorithms or approximated formulas and directly addresses the issue raised in [16].

B.4 Micro-geometric constraints

Similarly to the macro-geometric constraints, the micro-geometric constraints are listed in this subsection for completeness and reproducibility. These micro-geometric constraints are derived from formulas available in the ISO standards referenced in Appendix A.

Tooth profile modifications

The maximum tip relief or profile crowning depth is defined by the deflection of a rectangular beam with cross-section $b \times s$ under the maximum admissible load $F_{\max}$.

$$C_{\alpha a} = C_\alpha \leq \frac{4\, F_{\max} \left(({h_{aP}}^* + {h_{fP}}^*)\, m_t \right)^3}{E\, b\, s^3} \tag{55}$$

The tip relief start radius is defined by:

$$1.1\, r_{\text{active-contact}} \leq r_{c\ \alpha a} < r_a \tag{56}$$

Helix crowning

Maximum compression under $F_{\max}$, according to Hertz theory [47].

$$C_\beta \leq \frac{F_{\max}}{b} \left(\frac{1}{T_1 C} + \frac{1}{T_2 C} \right) \left(\frac{1-\nu_1^2}{E_1} + \frac{1-\nu_2^2}{E_2} \right) \tag{57}$$

Mitigation of manufacturing and assembly errors (contact recentering).

$$C_\beta \geq f_{H\beta T} + f_{\Sigma\delta}\, \cos(\alpha_{t_w}) + f_{\Sigma\beta}\, \sin(\alpha_{t_w}) \tag{58}$$

Profile slope tolerance

$$f_{H\alpha T} \leq (0.4\, m_n + 0.001\, d + 4) \left(\sqrt{2}\right)^{(A-5)} \tag{59}$$

where $1 \leq A \leq 12$ is the ISO quality class.

Helix form tolerance

$$f_{H\beta T} \leq \left(0.05\, \sqrt{d} + 0.35\, \sqrt{b} + 4\right) \left(\sqrt{2}\right)^{(A-5)} \tag{60}$$

B.5 Static transmission error computation procedure

Classical approaches for computing the static transmission error (STE) are based on an equation that models the static equilibrium of the gear pair at various positions of the driving wheel θ_1. The contact points are assumed to lie along theoretical contact lines, which are identified through a kinematic analysis of the system. The flexibility of the gear teeth is represented by a compliance matrix $\mathbf{H}(\theta_1)$, computed using a Ritz-Galerkin approximation. Contributions based on Hertz theory are added to this matrix to account for local deformations [48]. Tooth flank modifications and manufacturing errors are modeled as an initial gap $\mathbf{e}(\theta_1)$ between discretized contact lines. Additionally, misalignment and deviations caused by the global deformation of the entire gear train are considered in $e(\theta_1)$. The resulting contact problem, for each position θ_1 and a given transmitted load F, is formulated in matrix form as follows:

$$\begin{cases} \mathbf{H}(\theta_1)\,\mathbf{p}(\theta_1) = \delta(\theta_1)\,\mathbf{1} - \mathbf{e}(\theta_1) \\ \mathbf{1}^T\,\mathbf{p}(\theta_1) = F \end{cases} \tag{61}$$

with

$$\begin{cases} -\sum_j H_j(\theta_1)\,p_j(\theta_1) + \delta(\theta_1) \geq e_j(\theta_1) \\ p_j \geq 0 \end{cases} \tag{62}$$

In this constrained problem, the column vector $\mathbf{1}$ consists of components all equal to 1, while the column vector $\mathbf{p}$ represents the unknown distributed load, and the scalar function $\delta(\theta_1)$ corresponds to the unknown STE. The solutions $\mathbf{p}(\theta_1)$ and $\delta(\theta_1)$ are obtained by solving the equations using an optimization method.

The total computational workload was split into 20 parallel segments. Fourteen parts were run on Intel Xeon Haswell-based 16-core cluster nodes, with the remaining six distributed across three local machines: two parts on an AMD Ryzen 9 5900HS 8-core laptop, two on an Intel I7-11800H 8-core laptop, and two on an AMD Ryzen 9 5950X 16-core desktop. The total cumulative CPU time amounted to approximately 185.8 million core-seconds, equivalent to 51,610 hours or 2,150 days of single-core computation.

C Statistical characteristics of the dataset

In addition to Section 3, comprehensive statistical metrics of the 90 scalar features and targets composing the dataset are provided in Tables 6 and 7, including the mean, standard deviation, minimum, 25th percentile, median, 75th percentile, maximum, skewness, and kurtosis.

Table 6: Targets descriptive statistics.

Output Variable	Mean	StdDev	Min	P25	Median	P75	Max	Skew	Kurt
STE_0	49.49	55.32	0.00	16.10	32.57	60.98	599.89	3.07	13.42
STE_1	48.33	53.55	0.00	15.88	32.07	59.65	599.74	3.06	13.48
STE_2	47.31	52.21	0.00	15.66	31.57	58.40	599.57	3.06	13.61
STE_3	46.56	51.28	0.00	15.47	31.15	57.41	600.00	3.06	13.69
STE_4	46.11	50.63	0.00	15.36	30.94	56.94	599.95	3.06	13.72
STE_5	45.95	50.15	0.00	15.42	30.98	56.85	599.98	3.05	13.78
STE_6	45.99	49.79	0.00	15.58	31.19	57.05	599.97	3.05	13.84
STE_7	46.12	49.54	0.00	15.74	31.44	57.41	599.72	3.04	13.88
STE_8	46.24	49.34	0.00	15.87	31.62	57.71	599.73	3.03	13.89
STE_9	46.27	49.12	0.00	15.94	31.72	57.87	599.77	3.02	13.92
STE_10	46.20	48.82	0.00	15.98	31.75	57.89	599.43	3.01	13.99
STE_11	46.07	48.46	0.00	15.99	31.73	57.86	599.98	3.01	14.05
STE_12	45.94	48.12	0.00	15.98	31.68	57.85	599.97	2.99	14.00
STE_13	45.88	47.90	0.00	15.98	31.65	57.90	599.83	2.97	13.83
STE_14	45.95	47.90	0.00	16.01	31.69	58.08	599.97	2.96	13.68
STE_15	46.15	48.14	0.00	16.07	31.80	58.40	599.87	2.97	13.78
STE_16	46.46	48.56	0.00	16.14	31.98	58.83	599.61	3.00	14.18
STE_17	46.81	49.06	0.00	16.22	32.18	59.33	599.90	3.04	14.67
STE_18	47.13	49.51	0.00	16.28	32.37	59.79	599.99	3.06	14.98
STE_19	47.36	49.83	0.00	16.31	32.49	60.11	599.97	3.07	15.04
STE_20	47.48	50.00	0.00	16.30	32.55	60.28	599.92	3.06	14.91
STE_21	47.49	50.05	0.00	16.25	32.54	60.28	599.98	3.05	14.71
STE_22	47.44	50.05	0.00	16.18	32.49	60.20	599.89	3.04	14.54
STE_23	47.38	50.08	0.00	16.11	32.42	60.12	599.84	3.04	14.49
STE_24	47.38	50.17	0.00	16.04	32.37	60.09	599.95	3.04	14.52
STE_25	47.45	50.35	0.00	16.01	32.37	60.16	599.99	3.04	14.52
STE_26	47.59	50.60	0.00	16.00	32.41	60.33	599.98	3.04	14.39
STE_27	47.78	50.89	0.00	16.01	32.48	60.55	599.95	3.02	14.16
STE_28	47.97	51.16	0.00	16.01	32.55	60.79	599.98	3.00	13.87
STE_29	48.12	51.38	0.00	15.99	32.58	60.99	599.63	2.98	13.59
STE_30	48.19	51.52	0.00	15.94	32.56	61.09	599.98	2.96	13.36
STE_31	48.18	51.59	0.00	15.87	32.52	61.09	599.88	2.95	13.21
STE_32	48.13	51.61	0.00	15.79	32.45	61.05	599.98	2.95	13.16
STE_33	48.08	51.63	0.00	15.71	32.39	61.02	599.95	2.95	13.17
STE_34	48.06	51.70	0.00	15.64	32.36	61.03	599.77	2.95	13.17
STE_35	48.11	51.84	0.00	15.59	32.37	61.09	599.92	2.94	13.09
STE_36	48.23	52.07	0.00	15.55	32.40	61.22	599.90	2.93	12.92
STE_37	48.38	52.36	0.00	15.52	32.45	61.37	599.80	2.92	12.71
STE_38	48.54	52.67	0.00	15.49	32.49	61.53	599.98	2.91	12.52
STE_39	48.66	52.94	0.00	15.45	32.51	61.65	599.96	2.90	12.36
STE_40	48.73	53.14	0.00	15.41	32.49	61.72	599.99	2.90	12.25

Continued on next page

Table 6 – continued from previous page

Output Variable	Mean	StdDev	Min	P25	Median	P75	Max	Skew	Kurt
STE_41	48.75	53.28	0.00	15.36	32.44	61.72	599.82	2.89	12.15
STE_42	48.75	53.41	0.00	15.31	32.37	61.66	599.68	2.89	12.10
STE_43	48.77	53.61	0.00	15.28	32.31	61.60	599.98	2.90	12.12
STE_44	48.85	53.91	0.00	15.26	32.27	61.59	600.00	2.92	12.23
STE_45	49.00	54.34	0.00	15.26	32.28	61.68	599.98	2.94	12.40
STE_46	49.22	54.86	0.00	15.29	32.33	61.87	599.99	2.97	12.61
STE_47	49.49	55.42	0.00	15.34	32.39	62.11	599.95	2.99	12.81
STE_48	49.73	55.93	0.00	15.39	32.45	62.32	599.97	3.01	12.95
STE_49	49.91	56.32	0.00	15.43	32.48	62.45	599.97	3.03	13.00
STE_50	50.00	56.55	0.00	15.47	32.47	62.46	599.99	3.03	12.99
STE_51	50.02	56.63	0.00	15.50	32.44	62.38	599.96	3.03	12.96
STE_52	50.00	56.65	0.00	15.54	32.41	62.25	599.94	3.04	12.95
STE_53	50.04	56.73	0.00	15.59	32.42	62.18	599.99	3.04	13.00
STE_54	50.19	57.02	0.00	15.67	32.49	62.25	599.95	3.06	13.12
STE_55	50.50	57.59	0.00	15.78	32.63	62.51	599.97	3.09	13.40
STE_56	50.97	58.48	0.00	15.93	32.85	62.96	599.80	3.14	13.89
STE_57	51.54	59.58	0.00	16.10	33.11	63.52	599.93	3.22	14.60
STE_58	52.09	60.68	0.00	16.25	33.35	64.04	599.77	3.29	15.37
STE_59	52.47	61.44	0.00	16.39	33.54	64.37	599.99	3.34	15.90
STE_60	52.56	61.56	0.00	16.47	33.61	64.41	599.99	3.34	15.89
STE_61	52.28	60.84	0.00	16.48	33.55	64.07	599.99	3.29	15.30
STE_62	51.62	59.36	0.00	16.41	33.34	63.33	599.99	3.20	14.43
STE_63	50.65	57.38	0.00	16.28	33.01	62.25	599.95	3.12	13.71

Table 7: Features descriptive statistics.

Input Variable	Mean	StdDev	Min	P25	Median	P75	Max	Skew	Kurt
m_n	4.38	2.73	0.50	2.09	3.92	6.33	12.00	0.55	-0.60
alpha_n_deg	18.87	2.67	15.01	16.62	18.38	20.85	24.99	0.48	-0.84
beta_deg	6.91	9.23	0.00	0.00	0.00	13.59	29.93	1.04	-0.32
z1	91.64	35.34	11.00	63.00	93.00	122.00	150.00	-0.19	-1.01
z2	92.15	35.35	12.00	62.00	94.00	122.00	150.00	-0.18	-1.05
x_t1	0.05	0.53	-1.00	-0.38	0.07	0.48	1.00	-0.09	-1.04
haP_et1	1.27	0.37	0.50	0.99	1.29	1.56	2.00	-0.09	-0.89
hfP_et1	1.34	0.40	0.50	1.02	1.37	1.68	2.00	-0.25	-1.01
x_t2	-0.24	0.50	-1.00	-0.65	-0.31	0.12	0.98	0.46	-0.73
haP_et2	1.30	0.38	0.50	1.01	1.32	1.61	2.00	-0.10	-0.95
hfP_et2	1.32	0.40	0.50	1.01	1.33	1.66	2.00	-0.17	-1.01
b	43.44	30.14	2.83	20.04	36.45	60.33	166.04	0.95	0.51
a_w	411.86	285.03	18.98	174.08	362.56	584.78	1611.67	0.94	0.60
C_alphaCrowning1	19.99	55.91	0.00	0.00	0.45	15.63	1237.12	6.56	62.25
C_alphaCrowning2	31.38	82.43	0.00	0.00	0.63	23.97	1722.02	5.92	54.43
C_alphaTip1	18.50	53.54	0.00	0.00	0.00	13.76	1204.29	6.79	66.78
C_alphaTip2	28.97	78.53	0.00	0.00	0.00	20.88	1722.02	6.10	57.92
tipReliefStartRadius1	210.40	162.53	5.90	79.56	170.87	294.10	900.37	1.13	0.95
tipReliefStartRadius2	206.39	154.85	5.42	81.10	170.60	295.35	811.63	1.08	0.80
C_beta1	103.09	66.22	1.43	58.73	86.25	129.29	1077.50	2.31	11.65
C_beta2	103.09	66.25	5.22	58.77	86.34	129.29	1074.35	2.31	11.55
fH_alphaT1	6.16	3.89	0.00	2.94	5.88	8.84	20.74	0.44	-0.44
fH_alphaT2	6.16	3.88	0.00	2.94	5.88	8.84	19.96	0.43	-0.46
fH_betaT1	11.51	7.55	0.00	5.35	10.68	16.52	41.40	0.56	-0.26
fH_betaT2	11.49	7.48	0.00	5.35	10.72	16.49	39.78	0.54	-0.31
F	16792.32	25005.77	0.00	1631.33	6763.81	21184.36	284821.43	2.84	10.83

Target-to-target correlations. Correlation matrices were computed to assess linear relationships within and between input features and target variables, using the Pearson correlation coefficient r. The analysis of target variable correlations reveals a strong intrinsic structure within the STE signal. Adjacent STE components exhibit very high positive correlations, often exceeding $r = 0.95$, with correlation strength gradually decreasing as the distance between points along the time increases. This results in a banded pattern in the full correlation matrix, indicative of pronounced sequential dependencies. Such behavior, characteristic of time series data, suggests that multi-output modeling strategies or dimensionality reduction techniques may be particularly well-suited for this task.

Feature-to-feature correlations. Significant linear dependencies are present among several input features, as shown in Fig. 4. In particular, strong positive correlations ($r > 0.7$) are observed between the normal module m_n, facewidth b, operating center distance a_w, and the tip relief start radii ($r_{c\ \alpha a_1}$, $r_{c\ \alpha a_2}$). Very high correlations ($r > 0.9$) are specifically noted between the operating center distance and both tip relief start radii. These relationships arise from the fact that certain gear geometric parameters are linearly dependent, as it can be seen in Appendix B. Such inter-feature correlations suggest potential multicollinearity issues.

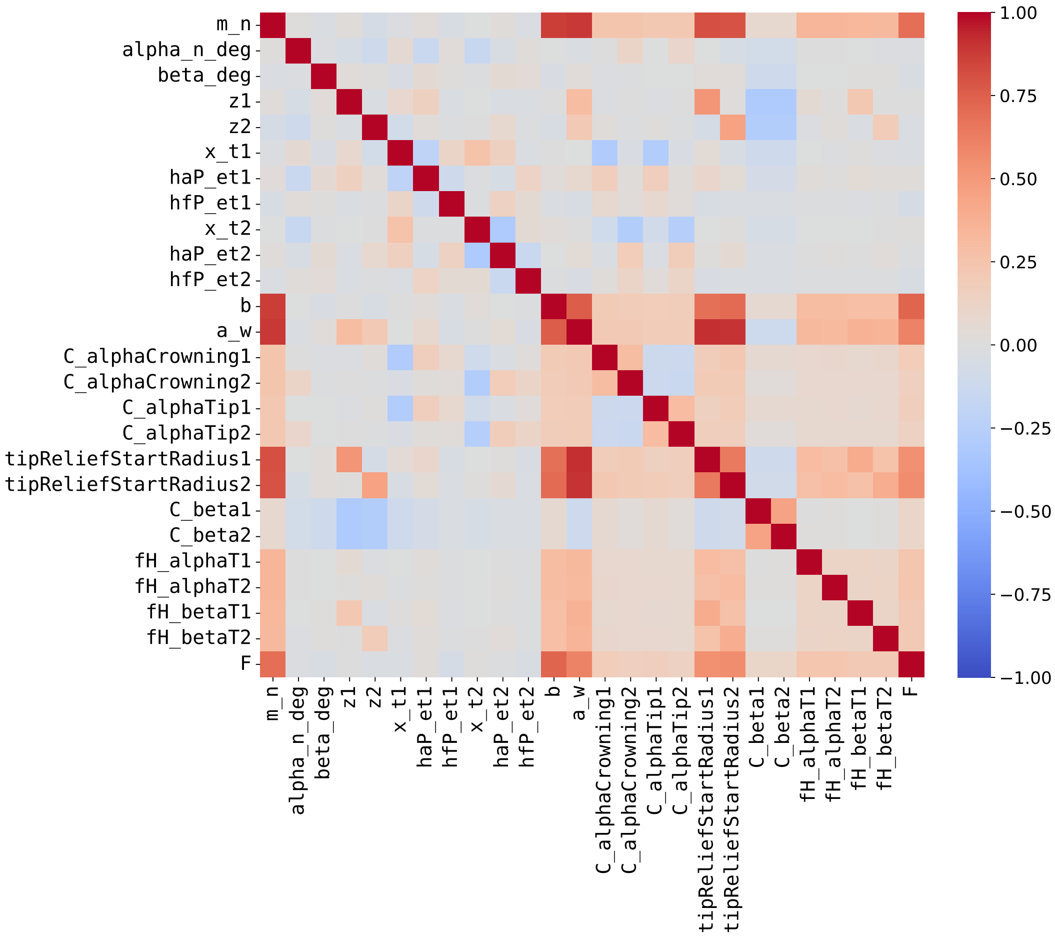


Figure 4: Feature-to-feature correlation matrix computed on the complete dataset with the Pearson correlation coefficient r.

Multicollinearity analysis of features. To formally quantify the linear dependencies among input features, Variance Inflation Factors (VIF) and eigenvalue analysis of the feature correlation matrix were performed. As presented in Fig. 5, the operating center distance a_w (VIF $\approx 5,900$), the tip relief start radius of gear 1 $r_{c\ \alpha a_1}$ (VIF $\approx 1,900$), and the tip relief start radius of gear 2 $r_{c\ \alpha a_2}$ (VIF $\approx 1,750$), all exhibit very high VIF values. The normal module m_n also shows a high VIF $\approx$ 20. The eigenvalues of the 26×26 feature correlation matrix range approximately from 1×10^{-4} to 5.58, leading to a condition number at approximately 53,600, confirming the strong multicollinearity.

Feature-to-target correlations. Moderate linear correlations between certain input features and the STE outputs, as reported in Fig. 6. For example, the normal module m_n shows correlations around 0.49, the transmitted force F ranges between 0.47 and 0.52, and the working center distance a_w exhibits correlations around 0.44 across many points of the STE profile. This is consistent with physical principles, as already addressed with the permutation importance of input features (see Fig. 3).

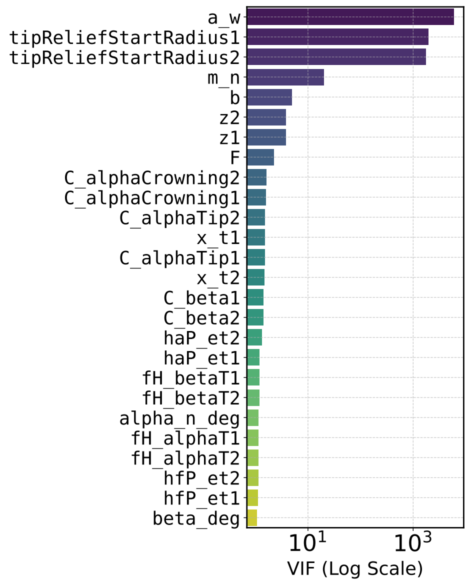


Figure 5: Variance Inflation Factors (VIF) for the features, computed on the complete dataset and plotted with a logarithmic x-axis.

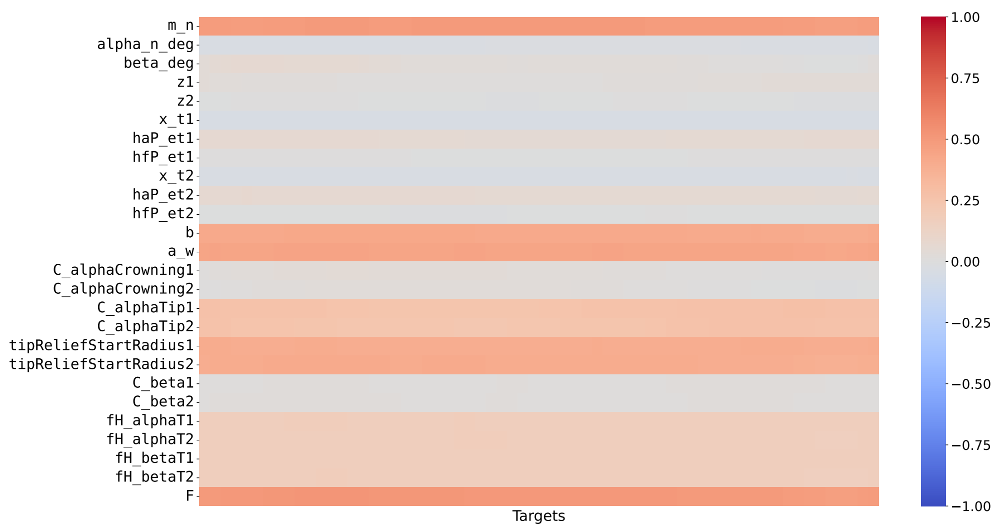


Figure 6: Feature-to-target correlation matrix computed on the complete dataset with the Pearson correlation coefficient r.

D How to run the surrogate model

D.1 Accessing datasets and model files

The surrogate model, named `Presto`, along with the complete dataset used for training, validation, and testing, is available at the following Google Drive link. This is a temporary hosting solution to preserve anonymity; a permanent repository will be provided later.

`https://drive.google.com/drive/folders/1mA_ucLHPQ1XancpWS8KDWxrSzdyPEtqP?usp=sharing`

D.2 Repository description

The repository contains the following structure (also described in the `readme.txt` file):

```
neurips2025_submission13213/
`-- presto_inference/
    |-- datasets/
    |   |-- presto_dataset.npz         # Complete dataset (31M samples)
    |   `-- test_subset_10k.npz        # Test subset for inference
    |-- presto_functions/              # Presto Python functions
    |   |-- __init__.py                # Presto Python package
    |   |-- inference.py               # Function for the inference pipeline
    |   |-- loading_utils.py           # Load metadata, model, scalers, data
    |   |-- model_defs.py              # MLP and Autoencoder class definitions
    |   |-- metrics.py                 # Metric functions
    |   `-- plotting.py                # Plotting results
    |-- presto_1.0.5_preview/          # Presto 1.0.5 Preview
    |   |-- presto_1.0.5_metadata.json # Model metadata
    |   `-- presto_1.0.5.safetensors   # Model
    |-- scalers/                       # Numpy scalers
    |   |-- scaler_X.joblib            # Features scaler
    |   `-- scaler_y.joblib            # Targets scaler
    |-- infer_presto.py                # Main inference script
    |-- license.txt                    # CC BY-NC-SA 4.0 License
    `-- readme.txt                     # Descritpion of the repository
```

D.3 Hardware and software requirements

Running `Presto` and loading the full dataset requires the following software environment:

- **Python version:** 3.9 or higher
- **Required libraries:**
  - `numpy`
  - `torch` (with CUDA support)
  - `scikit-learn`
  - `pandas`
  - `matplotlib`
  - `joblib`
  - `safetensors`
  - `pickle` (standard library)
  - `json` (standard library)
  - `os`, `time`, `gc`, `random`, `traceback` (standard libraries)
- **GPU:** An NVIDIA GPU is required.
- **CUDA toolkit:** A version compatible with the installed `torch` package. PyTorch must be installed with CUDA support (see `https://pytorch.org/get-started/locally/` for details).
- **RAM:** At least 32 GB of system memory is recommended to process the full dataset.

D.4 Open datasets

Two datasets are provided in the `datasets` folder. The first, `test_subset_10k.npz`, contains 10,000 randomly selected rows from the complete dataset and is intended for quick testing and inference. The second, `presto_dataset.npz`, contains the full dataset with 31 million rows. Users may place their own custom inference datasets in this folder to use them with the provided inference script.

Both datasets are stored as NPZ files and can be loaded with the following logic:

```python
import numpy as np

file_name = 'presto_dataset.npz' # File name

# Input keys (must be in this exact order)
input_keys = [
    'm_n', 'alpha_n_deg', 'beta_deg', 'z1', 'z2', 'x_t1',
    'haP_et1', 'hfP_et1', 'x_t2', 'haP_et2', 'hfP_et2',
    'b', 'a_w', 'C_alphaCrowning1', 'C_alphaCrowning2',
    'C_alphaTip1', 'C_alphaTip2', 'tipReliefStartRadius1',
    'tipReliefStartRadius2', 'C_beta1', 'C_beta2', 'fH_alphaT1',
    'fH_alphaT2', 'fH_betaT1', 'fH_betaT2', 'F'
]

np_data = np.load(file_name) # Load data
# Stack input features into matrix X
X = np.column_stack([np_data[key] for key in input_keys])
y = np_data['STE'] # Load target variable

# Data split ratios
VALIDATION_SIZE = 0.15
TEST_SIZE = 0.15
RANDOM_STATE_SPLIT = 42  # Seed for reproducibility
```

D.5 Running the Model

The model is run using the `infer_presto.py` inference script. Before launching the `Presto` inference, users must configure a few key parameters:

- **GPU selection**: Specify which GPU to use by setting the index value.

  ```
  TARGET_GPU_INDEX = 1 # By default TARGET_GPU_INDEX = 0
  ```

- **Inference Dataset**: Specify the path to the `.npz` file containing the input data. This file must at least contain the input features. Targets are optional and only required for `FULL_EVALUATION` mode.

  ```
  INFERENCE_NPZ_FILE = os.path.join(DATASET_FOLDER,
                                    "test_subset_10k.npz")
  ```

- **Script mode**: Choose between two modes:
  - `"FULL_EVALUATION"`: Loads inputs and reference targets, performs inference, computes evaluation metrics, and generates comparison plots.
  - `"INFERENCE_ONLY"`: Loads only inputs, performs inference, and generates prediction plots (no ground truth required).

  ```
  SCRIPT_MODE = "FULL_EVALUATION" # By default
  ```

- **Inference and evaluation configuration**:
  - `INFERENCE_BATCH_SIZE`: Sets the number of samples processed in parallel.
  - `N_SAMPLES_TO_PLOT`: Specifies how many samples to visualize.
  - `RANDOM_SEED`: Ensures reproducibility when selecting samples for plotting.

  ```
  INFERENCE_BATCH_SIZE = 10000 # By default
  N_SAMPLES_TO_PLOT = 20 # By default
  RANDOM_SEED = 42 # By default
  ```

`Presto` is executed with the command: `python infer_presto.py`.

Note: The inference script is designed for execution on a single GPU.

D.6 Model functions overview

The `Presto` model code includes the following key functions and classes:

- `run_inference_pipeline`: Runs the full inference workflow including scaling inputs, predicting with the model, and unscaling outputs. Returns predictions and timing information.
- `load_metadata`: Loads model architecture metadata from a JSON file.
- `load_scalers`: Loads pre-fitted input and output scalers used for data normalization.
- `load_combined_model`: Builds and loads the combined MLP and decoder model from configuration and safetensors weights.
- `load_inference_data`: Loads the input features for inference from an NPZ dataset file.
- `load_reference_data`: Loads optional reference target data for evaluation.
- `calculate_regression_metrics`: Computes common regression metrics including MSE, RMSE, MAE, R^2, and adjusted R^2.
- `calculate_relative_metrics`: Computes relative error metrics expressed as percentages of mean, range, and standard deviation.
- `MLP` (class): Defines the multilayer perceptron architecture used for the main prediction task.
- `Autoencoder` (class): Defines the autoencoder architecture used for decoding the MLP prediction.
- `plot_sample_comparisons`: Generates plots comparing predicted vs. reference STE for selected samples.
- `plot_predictions_only`: Generates plots showing predicted STE only.

All functions and classes are fully documented in their respective source files with detailed argument descriptions, outputs, and internal documentation.

D.7 Software environment used for the experiments

The experiments were conducted using the following software versions: Python 3.12.7, CUDA 12.7, PyTorch 2.5.1, scikit-learn 1.5.1, and Optuna 4.3.0.

E Limitations

While this work demonstrates the potential of deep learning for high-dimensional regression in mechanical engineering, several limitations must be acknowledged. Firstly, the dataset used for training is derived entirely from low-fidelity physics-based simulations. These simulations are computationally efficient and capture the dominant physical behaviors with reasonable accuracy, which result in computations of the STE with satisfactory qualitative and quantitative accuracy. However, they are based on simplifying assumptions, such as simplified contact modeling, which can result in discrepancies when compared to high-fidelity methods such as finite element analysis [49] or experimental measurements [50].

Secondly, while the proposed framework should be broadly applicable to any subfield of mechanical engineering where synthetic data can be generated efficiently, our evaluation is currently limited to gear mechanics. Gears were selected as a representative use case due to their ubiquity in mechanical systems and the statistical complexity of their design space. Nonetheless, validating the framework across a broader range of mechanical systems, particularly those governed by different physical principles, such as multiphysics systems, or data distributions, is an important direction for future work.

Finally, hardware constraints restricted the scale and complexity of the network architectures explored. The models were trained on only two consumer-grade GPUs, which imposed tight limits on the depth, width, and batch sizes of candidate architectures. As a result, the final surrogate represents a compromise between predictive performance and memory/computational constraints. Future work using more powerful hardware could explore larger models, train on larger datasets and possibly even attempt to reach double descent [24], to improve performance further.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The paper introduces a large-scale surrogate modeling framework for high-dimensional regression in mechanical engineering, backed by a 31M-sample dataset described in Sect. 3 and a 43M-parameter model described in Sect. 4.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The limitations are discussed in Appendix E.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [NA]

Justification: The paper is empirical and methodological.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: The full description of the model, hyperparameter optimizations and dataset are given in Sect. 4 and Appendix A, B and C. Moreover, both the complete dataset and model are provided in Appendix. D.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: Both the complete dataset and model are provided in Appendix D, with full instruction.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer: [Yes]

Justification: The detailed training parameters, model architectures, and optimization strategies are thoroughly described in Sect. 4.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: No error bars are reported, but the performance and error metrics of the model are provided in the Subsect. 4.4.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: The paper details GPU types, CPU specs, and training budgets, including compute times in Sect. 4.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The research conforms to the NeurIPS Code of Ethics. It does not involve human subjects, personal data, or potentially sensitive content. All data used in the study is synthetically generated using physics-based simulations, ensuring there are no privacy, consent, or fairness concerns. The work promotes transparency and reproducibility (through the release of code and data).

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [NA]

Justification: The paper focuses on foundational research in surrogate modeling for high-dimensional regression using synthetic, physics-based data in mechanical engineering. It does not involve human data, decision-making systems, or deployment in sensitive domains. As such, the work has no direct societal implications, positive or negative, at this stage. However, its potential downstream impact is primarily positive, enabling faster and more efficient engineering design workflows.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: The models and dataset presented in the paper pose no foreseeable risk of misuse. All data is synthetically generated through physics-based simulations without scraping, personal content, or generative capabilities. The surrogate model is domain-specific to mechanical engineering and cannot be easily repurposed for tasks with societal or ethical implications.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: All third-party assets used in this work, such as PyTorch, Optuna, and Scikit-learn, are properly cited and used in accordance with their respective open-source licenses. The dataset and models developed for this work are entirely original, synthetically generated in-house, and will be released under a CC BY-NC-SA 4.0 license for non-commercial and research purposes only.

Guidelines:

- The answer NA means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [Yes]

Justification: The paper introduces two new assets: a large-scale synthetic dataset generated from physics-based simulations and a trained deep neural network surrogate model. Both are released with accompanying documentation describing data generation methodology, feature definitions, usage instructions, and licensing terms. These assests are provided through a specific Google Drive link to ensure anonymity during the review process (see Appendix D).

Guidelines:

- The answer NA means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [NA]

Justification: The paper does not involve any crowdsourcing, human subjects, or data derived from human participants. All data is synthetically generated using a physics-based simulations, with no human interaction or annotation involved at any stage.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.

- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: This work does not involve human subjects or crowdsourcing in any form. All data is synthetically generated using physics-based simulations, and no ethical review or IRB approval is required.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

    Answer: [NA]

    Justification: This work does not involve Large Language Models (LLMs) in the development of core methods, experiments, or scientific contributions. The use of LLMs was limited to writing assistance and did not impact the originality, rigor, or methodology of the research.

    Guidelines:

    - The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.